Git

Git is a distributed version control system

Version Control System (VCS) or Source Code Manager (SCM): A VCS allows you to: revert files back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

Three of the most popular version control systems are:

• Git

• Subversion

• Mercurial

There are two main types of version control system models:

the centralized model - all users connect to a central, master repository

the distributed model - each user has the entire repository on their computer

A version control system (abbreviated as VCS) is a tool that manages different versions of source code. A source code manager (abbreviated as SCM) is another name for a version control system.

The process is add new file/modify existing file, move the changes to the staging index and commit the changes to the repository.

Commit (snapshot): Git thinks of its data like a set of snapshots of a mini file system. Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.

Repository (repo): A directory that contains your project work, as well as a few files (hidden by default in Mac OS X) which are used to communicate with Git. Repositories can exist either locally on your computer or as a remote copy on another computer.

Working Directory: The files that you see in your computer's file system. When you open your project files up on a code editor, you're working with files in the Working Directory.

This is in contrast to the files that have been saved (in commits!) in the repository.

When working with Git, the Working Directory is also different from the command line's concept of the current working directory which is the directory that your shell is "looking at" right now.

Checkout: When content in the repository has been copied to the Working Directory. It is possible to checkout many things from a repository; a file, a commit, a branch, etc.

Staging Area or Staging Index or Index: A file in the Git directory that stores information about what will go into your next commit. You can think of the staging area as a prep table where Git will take the next commit. Files on the Staging Index are poised to be added to the repository.

SHA: A SHA is basically an ID number for each commit. It is a 40-character string composed of characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git. "SHA" is shorthand for "SHA hash". A SHA might look like this: e2adf8ae3e2e4ed40add75cc44cf9d0a869afeb6

Branch: A branch is when a new line of development is created that diverges from the main line of development. This alternative line of development can continue without altering the main line. Going back to the example of save point in a game, you can think of a branch as where you make a save point in your game and then decide to try out a risky move in the game. If the risky move doesn't pan out, then you can just go back to the save point. The key thing that makes branches incredibly powerful is that you can make save points on one branch, and then switch to a different branch and make save points there, too.

zsh—version – what version of GIT

Required Commands

Heads up! We'll be using the following terminal commands in this lesson:

ls - used to list files and directories

mkdir - used to create a new directory

cd - used to change directories

rm - used to remove files and directories

Git Init Recap

Use the git init command to create a new, empty repository in the current directory.

$ git init

Running this command creates a hidden .git directory. This .git directory is the brain/storage center for the repository. It holds all of the configuration files and directories and is where all of the commits are stored.

Git Clone Recap

The git clone command is used to create an identical copy of an existing repository.

$ git clone <path-to-repository-to-clone>

This command:

* takes the path to an existing repository
* by default will create a directory with the same name as the repository that's being cloned
* can be given a second argument that will be used as the name of the directory for example: git clone <path-to-repository-to-clone> NewNameOf Repository
* will create the new repository inside of the current working directory

$ git status

The git status command will display the current status of the repository. It will tell us what Git is thinking and the state of our repository as Git sees it.

The git log command is used to display all of the commits of a repository.

$ git log

By default, this command displays:

* the SHA
* the author
* the date
* and the message

$ git log --oneline - command with just the short SHA and the commit message.

This command:

* lists one commit per line
* shows the first 7 characters of the commit's SHA
* shows the commit's message

$ git log --stat

This command:

* displays the file(s) that have been modified
* displays the number of lines that have been added/removed
* displays a summary line with the total number of modified files and lines that have been added/removed

git log -p

To recap, the -p flag (which is the same as the --patch flag) is used to alter how git log displays information:

* displays the files that have been modified
* displays the location of the lines that have been added/removed
* displays the actual changes that have been made

git log -p –stat - will also show the stats info above the patch info.

git log -p -w - will show the patch information, but will not highlight lines where only whitespace changes have occurred.

$ git log -p fdf5493 – first 7 numbers of SHA

By supplying a SHA, the git log -p command will start at that commit! No need to scroll through everything! Keep in mind that it will also show all of the commits that were made prior to the supplied SHA.

$ git show fdf5493

The git add - command is used to move files from the Working Directory to the Staging Index.

$ git add . – move all files in working directory

$ git add <file1> <file2> … <fileN>

This command:

* takes a space-separated list of file names
* alternatively, the period . can be used in place of a list of files to tell Git to add the current directory (and all nested files)

$ git rm --cached <file>... to unstage, but not removing file from directory;

$ git commit – take files from the staging index and save them in the repository;

$ git commit -m "Initial commit" – commit with a “commit message”

$ git diff command is used to see changes that have been made but haven't been committed. This command displays:

* the files that have been modified
* the location of the lines that have been added/removed
* the actual changes that have been made

the git tag -a beta(version) - command is used to add a marker on a specific commit. The tag does not move around as new commits are added. This command will:

* add a tag to the most recent commit
* add a tag to a specific commit if a SHA is passed

$ git tag -d v1.0 - delete tag (v1.0 – version of tag)

$ git tag -a v1.0 a87984 – adding tag for past commits with SHA number at the end;

$ git log –decorate – to check the tag;

$ git branch - command is used to interact with Git's branches:

* list all branch names in the repository
* create new branches
* delete branches

$ git branch (name of the new branch example: sidebar) – creating new branch with a name